2013

Evaluations of heuristic algorithms for teamwork-enhanced task allocation in mobile cloud-based learning

Geng Sun
University of Wollongong, gs147@uowmail.edu.au

Jun Shen
University of Wollongong, jshen@uow.edu.au

Junzhou Luo
Southeast University

Jianming Yong
University Of Southern Queensland

Publication Details

Sun, G., Shen, J., Luo, J. & Yong, J. (2013). Evaluations of heuristic algorithms for teamwork-enhanced task allocation in mobile cloud-based learning. International Conference on Computer Supported Cooperative Work in Design (pp. 299-304). Australia: IEEE.

Evaluations of heuristic algorithms for teamwork-enhanced task allocation in mobile cloud-based learning

Abstract
Enhancing teamwork performance is a significant issue in mobile cloud-based learning. We introduce a service oriented system, Teamwork as a Service (TaaS), to realize a new approach for enhancing teamwork performance in the mobile cloud environment. To coordinate most learners' talents and give them more motivation, an appropriate task allocation is necessary. Utilizing the Kolb's learning style (KLS) to refine learner's capabilities, and combining their preferences and tasks' difficulties, we formally describe this problem as a constraint optimization model. Two heuristic algorithms, namely genetic algorithm (GA) and simulated annealing (SA), are employed to tackle the teamwork-enhanced task allocation, and their performances are compared respectively. Having faster running speed, the SA is recommended to be adopted in the real implementation of TaaS and future development.

Keywords
heuristic, cloud, evaluations, task, learning, allocation, enhanced, mobile, teamwork, algorithms

Disciplines
Engineering | Science and Technology Studies

Publication Details
Sun, G., Shen, J., Luo, J. & Yong, J. (2013). Evaluations of heuristic algorithms for teamwork-enhanced task allocation in mobile cloud-based learning. International Conference on Computer Supported Cooperative Work in Design (pp. 299-304). Australia: IEEE.

Evaluations of Heuristic Algorithms for Teamwork-Enhanced Task Allocation in Mobile Cloud-Based Learning

Geng Sun, Jun Shen
Faulty of Informatics, University of Wollongong
Wollongong, NSW, Australia
gs147@uowmail.edu.au, jshen@uow.edu.au

Junzhou Luo
School of Computer Science and Engineering, Southeast University
Nanjing, China
jluo@seu.edu.cn

Jianming Yong
Faculty of Business and Law, University of Southern Queensland
Toowoomba, QLD, Australia
yongj@usq.edu.au

***Abstract*—Enhancing teamwork performance is a significant issue in mobile cloud-based learning. We introduce a service oriented system, Teamwork as a Service (TaaS), to realize a new approach for enhancing teamwork performance in the mobile cloud environment. To coordinate most learners' talents and give them more motivation, an appropriate task allocation is necessary. Utilizing the Kolb's learning style (KLS) to refine learner's capabilities, and combining their preferences and tasks' difficulties, we formally describe this problem as a constraint optimization model. Two heuristic algorithms, namely genetic algorithm (GA) and simulated annealing (SA), are employed to tackle the teamwork-enhanced task allocation, and their performances are compared respectively. Having faster running speed, the SA is recommended to be adopted in the real implementation of TaaS and future development.**

Keywords—Mobile cloud-based learning, teamwork-enhanced, Kolb's learning style, task allocation, heuristic alogrithms

I. INTRODUCTION

More and more mobile devices are able to access contents and resources in the learning management systems (LMSs), hence mobile learning (m-learning) becomes a novel trend of electronic learning (e-learning) so that learners can freely do learning activities wherever they are and whenever they want [1]. On the other hand, cloud computing emerges to change the traditional system hosting method with its advantages in massive data handling, large storage and on-demand utilizing. Consequently, migrating current LMSs to cloud or directly developing them over cloud provide many conveniences to solve the deployment and operation issues in e-learning. The definition of m-learning has also been evolved by embracing it to cloud computing, where learners are free to use these cloud-hosting LMSs through mobile devices. This is a new learning style, namely mobile cloud-based learning [2].

Practitioners believe that mobile cloud-based learning benefits learners in many aspects of collaborative interactions, rather than just using their mobile devices only for inputting and outputting. However, it still lacks mechanism for enhancing teamwork performance in mobile environment. The mobile cloud-based learning context is quite different from that of traditional learning. Teams in mobile learning are more focused on task-related outcomes, similar to the common distributed teams, which are sometimes called virtual teams [3]. Considering such requirements, and other issues negatively affecting team learning, in [4], we introduced a new approach based on orchestrating several web services to execute a rational learning flow, which enhances teamwork performance in mobile cloud environment. These web services, named teamwork as a service (TaaS), are designed to work as a whole system. It plays like a third-party system to add functions to current cloud-hosting LMSs thereby learners can follow the executions of the related services flow and also refine their team learning activities accordingly.

Because interpersonal interactions in mobile cloud-based learning are complex to maintain, especially there are less face-to-face communications, therefore the optimal task allocations are necessary to avoid confusion and misunderstanding in the teamwork process. Kolb's learning style (KLS) is a classical educational theory that identified four learning styles in the team learning, namely accommodating, assimilating, converging and diverging [5][6]. It is important to evaluate learners' capabilities with regard to these four learning styles when system attempts to assign the most appropriate tasks to them. One the other hand, learners' comprehensive teamwork skills, preferences and the difficulties of tasks also need to be taken into consideration.

The main contribution of this paper is to present a constructive approach of task allocation in mobile cloud-based learning, using KLS to accurately allocate responsible tasks to each learner in order to enhance teamwork performance. We employ two heuristic algorithms, Genetic Algorithm (GA) and Simulated Annealing (SA), to facilitate the task allocation, the performances of which are compared to support real development in mobile cloud environment.

The rest of the paper is organized as follows. Section II illustrates the system framework of TaaS. Section III models the mobile cloud-based learning scenario and KLS, and describes some attribute setting for task allocation. Section IV presents the GA and SA algorithms and the related

experimental results are shown in Section V. Section VI concludes our work and points out what further research may focus on.

II. SYSTEAM FRAMEWORK

Kolb team learning experiences (KTLE), which has seven modules, guides how to form a team in a sequential order to improve team learning [7]. Following its guidance, we designed the TaaS to work in conjunction with cloud-hosting LMSs, as a third-party system for functional supplements. There are frequent interactions and instant information synchronizations between TaaS and cloud-hosting LMSs, both of which are mobile accessible. TaaS consists of five web services, each of which takes the functions of one or more modules in the KTLE and they altogether execute to realize a teamwork-enhanced learning flow. New step in the learning flow can be triggered immediately when previous event finishes or a message jumps in.

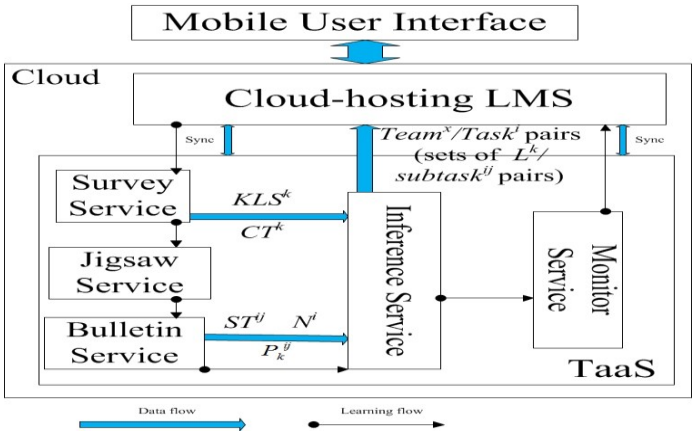

Figure 1. System Framwork of TaaS

As shown in the Figure 1, once the topics of team assignments are released, learners and teachers login the TaaS use their validated LMS accounts. The single-sign-on (SSO) technique is used to support the login process. The first information synchronization between TaaS and LMS is triggered, and both of them will share the same user information in the whole team learning process. Note the symbols on the connections will be explained in later sections while main modules are discussed as follows.

Knowing each other is a significant step that can help teammates to get ready for their following works. The Survey Service provides a platform for the "introduction to the teams". It offers interfaces to learners for answering questions to investigate their capabilities, which are about KLS and comprehensive teamwork skills. The survey is single-choice based to adapt the limitation of screen sizes and typing methods of mobile devices, and it can be operated as self-assessment or peer-assessment. The questions of surveys come from [8] [9] [10].

Jigsaw classroom is widely used in team learning for deepening learners' understanding to the learning content [11]. Borrowing this idea, we use the Jigsaw Service to form a cloud jigsaw classroom that works to help learner to understand the "team purpose".

Next, during the Bulletin Service's execution, learners are free to build their conceptions and schedule their schemes for drawing the outline of the "team context". As a requirement, a pre-planned task should meet the expectation that it is suitable for the workload of an imaginary team, which consists of several subtasks and detailed steps of them, whether offered by an original team in cloud jigsaw classroom or by an individual learner. Every task is alternative and potential to be adopted as a team's assignment in the final task allocation. To specify its difficulty, the assignment publishers are asked to give every subtask expected-achievable values with regard to the four aspects of the KLS, in order to mark each of them to be better completed by a leaner who has the appropriate capabilities. Moreover, each learner is encouraged to give a preference grade to every published subtask when browsing the bulletin.

The core of TaaS is the Inference Service. It takes the data recorded in the Survey service and the Bulletin Service as raw information, and uses certain rules to make a unique rational decision what the "team membership" and "team roles" are. In the ultimate task allocation, each subtask is assigned to one learner, meanwhile learners who take subtasks belonged to the same task will be grouped into the same team.

After the task allocation is finished, TaaS synchronizes the team information with LMS. Then learners participate in "team process" and "team action". The Monitor Service is designed to support mutual supervision in these (two) stages.

III. TEAMWORK-ENHANCED TASK ALLOCATION MODEL

A. Learner and Task Modeling

Suppose there are a number of learners using the TaaS for enhancing their teamwork performance when being involved mobile cloud-based learning. Let the L^k denotes the k^{th} learner. In the Survey Service, L^k's capability will be compiled from questionnaires, from both self-assessment and peer-assessment. There are five sets of questionnaires, and the results of each will be recorded in a matrix, in which each column stands for a question while each row corresponds to a learner who gives the marks. So five matrixes are obtained, they are $\{AC^k\}$, $\{AS^k\}$, $\{C^k\}$, $\{D^k\}$ and $\{CT^k\}$. For example, the capability of accommodating (AC) of L^k can be stated as:

$$\{AC^k\} = \begin{pmatrix} M_1^1 & M_1^2 & \dots & M_1^n \\ M_2^1 & M_2^2 & \dots & M_2^n \\ \dots & \dots & \dots & \dots \\ M_m^1 & M_m^2 & \dots & M_m^n \end{pmatrix} \quad (1)$$

Where: M_m^n means the mark for the n^{th} question of the accommodating aspect, which is given in the m^{th} assessment, and M_m^n is an integer between 0 and 10. The n depends on the question title's order and the m is in accordance with sequence of questionnaire submission times.

In this matrix $\{AC^k\}$, means of each column describe strengths of different types of accommodating, and we use the

next equation to calculate the value of accommodating capability of L^k:

$$AC^k = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} M_j^i}{nm} \quad (2)$$

In the same way, the Survey Service calculates the values for the other four matrixes. Hence, we got these values: AS^k, C^k, D^k and CT^k. They represent the capability values of assimilating, converging, diverging and comprehensive teamwork skills, respectively. Therein, we let a 4-tuple $KLS^k = \{AC^k, AS^k, C^k, D^k\}$ denote the KLS capability values of L^k according to that they are closely related.

In the Bulletin Service, a published $subtask^{ij}$ represents it is the j^{th} subtask of the i^{th} task. Its expected-achievable values are set in a 4-tuple ST^{ij}, where $ST^{ij} = \{AC^{ij}, AS^{ij}, C^{ij}, D^{ij}\}$, each value is a real between 1 and 10.

The variable P_k^{ij} denotes the preference grade of the $subtask^{ij}$, given by the k^{th} learner. Note the P_k^{ij} is an integer between 1 and 5, the higher the grade is, the more preferred by the learner to do a subtask. Typically we can assume that there are five types of subtasks, which are in turn regarded as "very interesting", "interesting", "ordinary", "uninteresting" and "very uninteresting" if they separately got the preference grade 5, 4, 3, 2, 1, by a specific learner.

B. Problem Description

Suppose in a possible task allocation, the learner L^k is allocated with the $subtask^{ij}$, it is necessary to check whether they are roughly matching and on which level they suit to each other. We introduce two attributes to describe the deviation of that learner versus subtask. The first one is *DeP*, which stands for the preference gap between learner's ideal and reality, where:

$$DeP_k^{ij} = 5 - P_k^{ij} \quad (3)$$

And the second one is *DeK*, which denotes the deviation of learner's KLS capability values versus a subtask's expected-achievable values, where:

$$DeK_k^{ij} = -\{sign[\sum(KLS^k - ST^{ij})]\} \cdot \| KLS^k - ST^{ij} \| \quad (4)$$

Subject to:

$$KLS^k - ST^{ij} = \{AC^k - AC^{ij}, AS^k - AS^{ij}, C^k - C^{ij}, D^k - D^{ij}\} \quad (5)$$

$$\| KLS^k - ST^{ij} \| = \sqrt{(AC^k - AC^{ij})^2 + (AS^k - AS^{ij})^2 + (C^k - C^{ij})^2 + (D^k - D^{ij})^2} \quad (6)$$

Both of these deviations are the lower the better. An ideal DeK_k^{ij} is below 0.

The basic idea of the task allocation is to assign learners with their appropriate subtasks. However, it may result in a situation where the chosen subtasks cannot compose into full tasks. For example, there are two tasks, each consists of three subtasks, but the Inference Service allocates two subtasks of each to four best-suited learners. In this situation, teams cannot be formed. Moreover, in team learning, it cannot start with the condition of learners having got their individual subtasks beforehand, as they still need to be grouped into teams. To enhance teamwork performance, we need to consider the whole strength of a team when grouping them. Furthermore, if suitable, it is possible that two or more teams are assigned the same task as their assignments. To avoid misunderstanding, we use a variable *x* to mark a team tag. Sums of *DeP*, *DeK* and *CT* in a potential team *x* can be stated as:

$$^xDeP^i = \sum {}^xDeP_k^{ij} \quad (7)$$

$$^xDeK^i = \sum {}^xDeK_k^{ij} \quad (8)$$

$$^xCT = \sum {}^xCT^k \quad (9)$$

N^i denotes the number of subtasks in the $task^i$.

We will separately discuss features of two scenarios of forming a team.

1) "keeping the balance between each team"

It means that if we regard each upcoming team as an independent unit, its integrated comprehensive teamwork skills, preferences, and capability values are highly close to those of other units. Therefore, we can deem that the inter-team competitions between the upcoming teams start from the same scratch line and are supposedly fair.

Briefly, each upcoming team should have the nearly equal ^xCT, followed by the respectively proximate $^xDeP^i$ and $^xDeK^i$.

2) "Letting the learners to show their capabilities mostly"

It means each of them is able to take advantage of their superiorities as much as possible, so that whether the team members are "good at" and "happy to do" their upcoming subtasks will be the main indexes that supervise the reasoning processing of task allocation.

That is to say, each upcoming team's $^xDeP^i$ and $^xDeK^i$ should be minimized. Under this premise, the ^xCT level between teams is better to be kept in balance as possible.

As the Inference Service is part of the TaaS working for assisting real mobile-based cloud learning, several situations should be considered realistically. The ultimate purpose of each learner who participates in the cloud-based course is to get a final grade for their team assignment, in order to pass the subject. So in the task allocation, no learners should be left out, though they might have unsatisfactory capabilities or unexpected performance. On the other hand, overflowing subtasks, which results in the unshaped team, is not allowed or encouraged. An integrated task should be allocated to a team rather than just part of its subtasks being allocated to several learners.

IV. ALGORITHMS

The scale of solution spaces of the teamwork-enhanced task allocation is *k*!, where *k* is the number of learners. We attempt to use heuristic algorithms to tackle the problem out.

In this section, we describe, the details of using genetic algorithm (GA) and simulated annealing (SA) to solve the problem of teamwork-enhanced task allocation. To simulate the real scene of mobile cloud-based learning, which is large-scale and distributed, we suppose the number of learners and tasks/subtasks are big enough.

A. Genetic algorithm method

GA is an optimal self-adaptive heuristic algorithm, which simulates the natural biological selection and genetic evolution mechanism. The basic idea of GA is inspired by evolution process in the natural world, to optimize candidate solutions towards better ones [12] [13]. Traditionally, candidate solutions start randomly and change in generations, by selection, crossover and mutation. Every generation is evaluated by a fitness function and the new generation is then used in the next iteration of the algorithm. Once a satisfactory of fitness level has been reached, the iterations terminate and the algorithm outputs the final generation as the optimal solution.

To start the GA operation, arrays of k learner/subtask pairs are randomly generated, where k is the number of learners. In each array, the integrities of tasks should be checked. If existing any overflowing subtask, that array will not be adopted as the initial solution. Taking these initial solutions as individuals (chromosomes), we need to encode them into populations (genomes) for creating the first generation.

A fitness function transfers the task allocation from multi-objective optimization to single-objective optimization. For the first scenario mentioned in Section III, to make the proximate ${}^{x}CT$, ${}^{x}DeP^{i}$ and ${}^{x}DeK^{i}$ between teams, total teams' variances of these parameters should be respectively minimized. However, for each attribute, several solutions may have different means but with the similar variances. A special situation is that the original difference of potential teams is little. To avoid the evaluation blindly terminates in a partial balance, we take minimizing the means of the *DeP* and the *DeK* of all teams into consideration. So we use the next equation as the fitness function:

$$R_m=\alpha\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{{}^{x}CT^{i}}{N^{i}}-\overline{CT}\right)^2}+\beta\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{{}^{x}DeP^{i}}{N^{i}}-\overline{DeP}\right)^2}+\gamma\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{{}^{x}DeK^{i}}{N^{i}}-\overline{DeK}\right)^2}+\varepsilon\overline{DeP}+\eta\overline{DeK} \tag{10}$$

For the second scenario, in a candidate solution, minimizing the total *DeP* and *DeK* is more important than minimizing the variance of *CT*. so we take the next fitness function:

$$R_m=\alpha\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{{}^{x}CT^{i}}{N^{i}}-\overline{CT}\right)^2}+\sum_{i=1}^{n}\left(\beta\,{}^{x}DeP^{i}+\gamma\,{}^{x}DeK^{i}\right) \tag{11}$$

where each Greek letter in (10) and (11) represents the weight for that attribute.

The aim of selection operator is to remove the poor solution with higher fitness. Then the selected individuals evolve to the next generation through the effect of crossover operator and mutation operation. We choose the top percent selection as the selection operator, the partially matched crossover as the crossover operator and the uniform mutation as the mutation operator. Let the population size is *2k*. The pseudo code of GA is shown below:

The pseudo code of GA

Input: $KLS^{k}, CT^{k}, ST^{ij}, P_{k}^{ij}, N^{i}$

Output: $Team^{x}/Task^{i}$ pairs (sets of $L^{k}/subtask^{ij}$ pairs)

begin: Calculate *DeP, DeK, CT*.

Randomly generate arrays of k $L^{k}/subtask^{ij}$ pairs

Check the task integrity in each array, give up unmatched ones.

Take the matched individuals as the initial population. Make the population size as *2k*.

for each individual $\in$ population **do**

Evaluate the fitness of each individual using R_m.

end for

while iteration times < max iteration time **do**

Select the individuals with lower fitness.

Use crossover operator to produce offspring.

Operate offspring through mutation operator.

Evaluate the fitness of new individuals using R_m.

Take the lower-fitness individuals to replace the old ones.

end while

Output the task allocation.

end

B. Simulated annealing method

SA is a generic heuristic algorithm for locating a good approximation to the global optimization problem in a large scale. It borrows the idea from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects [14].

The initialization of SA is similar to that of GA. The initial solution set is formed by numbers of randomly generated candidate solutions, each of which is an array of k learner/subtask pairs. Certainly, the integrities of tasks should also be checked. Let the initial set include *2k* matched candidate solutions.

The operation of SA includes two loops, namely inner loop and outer loop. In the inner loop, an objective function is defined as same as the fitness function R_m in GA, (10) for the first scenario and (11) for second scenario, respectively. The target of objective function is using R_m to evaluate each solution in order to obtain the calculation result, namely energy (E), which is also called fitness in GA. In a candidate solution, 2 learner/subtask pairs are randomly selected, and their positions of learners are swapped, in order to generate a new solution. The energy of current solution ($E_{current}$) and new solution (E_{new}) should be evaluated by R_m. Then we take the Metropolis Criterion as reference for accepting new solution. The acceptance probability (AP) can be stated as:

$$AP=\begin{cases}1 & ,E_{new}<E_{current}\\ \exp\dfrac{E_{new}-E_{current}}{t_{current}} & ,E_{new}>E_{current}\end{cases} \tag{12}$$

where $t_{current}$ is the value of current temperature parameter. The inner loop terminates at the condition of that the energies of the optimal solution in 5 continuous new solution sets

($E_{optimal}$) vary in a very narrow range. To mark the range clearly, we let the variance of these 5 continuous energies less than 0.001.

In the outer loop, the initial temperature (t_0) should be high enough to allow acceptance of any energy moving. We set t_0 =100. A cooling strategy is used to update the previous temperature parameter t by multiplying a cooling schedule incremental multiplier λ, so:

$$t_{i+1} = \lambda \cdot t_i \tag{13}$$

where $0 \le \lambda \le 1$. If the temperature decreases too fast, the algorithm may be trapped in local minimum [15]. Hence, we claim a useful value 0.95 as the λ in this paper.

There are two alternative termination conditions of the outer loop. Firstly, the parameter t meets the lowest temperature (t_{stop}), which is 10^{-7} in this paper. Secondly, the optimal solutions searched by SA do not change obviously for continuous times, which means, as we set, the variance of 5 continuous energies is less than 0.001. The final solution is outputted once one of these two conditions occurs.

The pseudo code of SA is shown below:

The pseudo code of SA

Input: $KLS^k, CT^k, ST^{ij}, P_k^{ij}, N^i$

Output: $Team^x/Task^j$ pairs (sets of $L^k/subtask^{ij}$ pairs)

begin: Calculate *DeP, DeK, CT.*
  Randomly generate arrays of k $L^k/subtask^{ij}$ pairs
  Check the task integrity in each array, give up unmatched ones.
  Take the matched solutions as the initial solution sets. Make the set size as $2k$.
  $t = t_0$
 while current temperature $t >$ lowest temperature t_{stop} **do**
  // outer loop
  for each solution $\in$ solution set **do** // inner loop
   evaluate the energy of current solution ($E_{current}$) using R_m
   choose two learner/subtask pairs
   swap the position of learners to produce new solution
   evaluate the energy of new solution (E_{new}) using R_m
   accept new solution based on acceptance probability AP
   select the optimal solution in the solution set
   evaluate its energy ($E_{optimal}$)
    if variance of 5 continuous $E_{optimal} < 0.001$
     break // terminate inner loop
    end if
  end for
  select the optimal solution in the solution set
  evaluate its energy ($E_{optimal}$)
   if variance of 5 continuous $E_{optimal} < 0.001$
    break // terminate outer loop
   end if
  $t = \lambda t$ // cooling
 end while
  Output the task allocation.
end

V. Experiment Results and Comparison

In this section, we present the experiment results of teamwork-enhanced task allocation by GA and SA, and compare their performances. Both the algorithms are implemented in Matlab, running on a laptop with 2.40 GHz Intel Core i5 CPU and 4GB memory.

Firstly, we determine that these two algorithms make the task allocation feasible. The data of learner and task information with all attributes are randomly simulated by Matlab, obeying normal distribution. For the function R_m, we set the weights, in the first scenario, α=0.5, β=0.15, γ=0.25, ε=0.05, η=0.05, and, in the second scenario, α=0.2, β=0.4, η=0.4. For GA, we set the crossover probability is 0.9, meanwhile the mutation probability is 0.2. For both GA and SA, the number of learners (k) and subtasks are separately chosen as 100 and 200.

Having met the terminal condition, the algorithm outputs solutions, including 100 learner/subtask pairs, for allocating learners to their most appropriate subtasks. Both the algorithms can give the results as we predicted. For example, the output of GA is shown in Figure 2. In the first scenario, we can find that learners are divided into 20 teams and the values of total *CT*, *DeP* and *DeK* of each team are separately balanced on nearly the same levels. That is to say, the three attributes between teams are all in close proximities, which mean that the teams have almost equal capabilities and preferences to achieve goals of their responsible tasks. And in the second scenario, as the solution would group learners into 23 teams, the *DeK* attributes of each team are below 0, so that each team is competent to their allocated tasks. The result that the *DeP* level of each team is less than 4, is due to the team size is 4 to 6 persons, that means the allocated tasks are enjoying high preferences as being deemed better than "interesting".

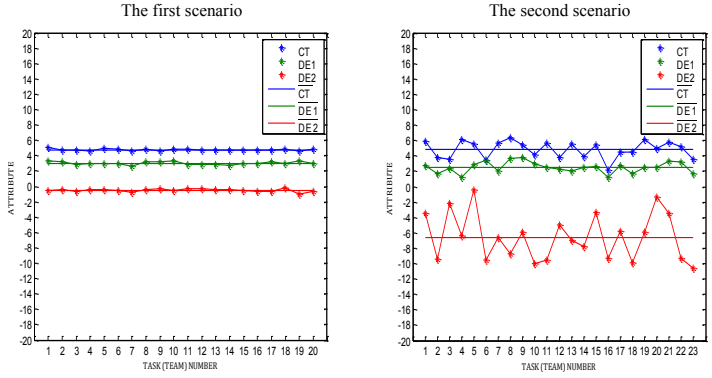

Figure 2. Task allocation for two scenarios by GA

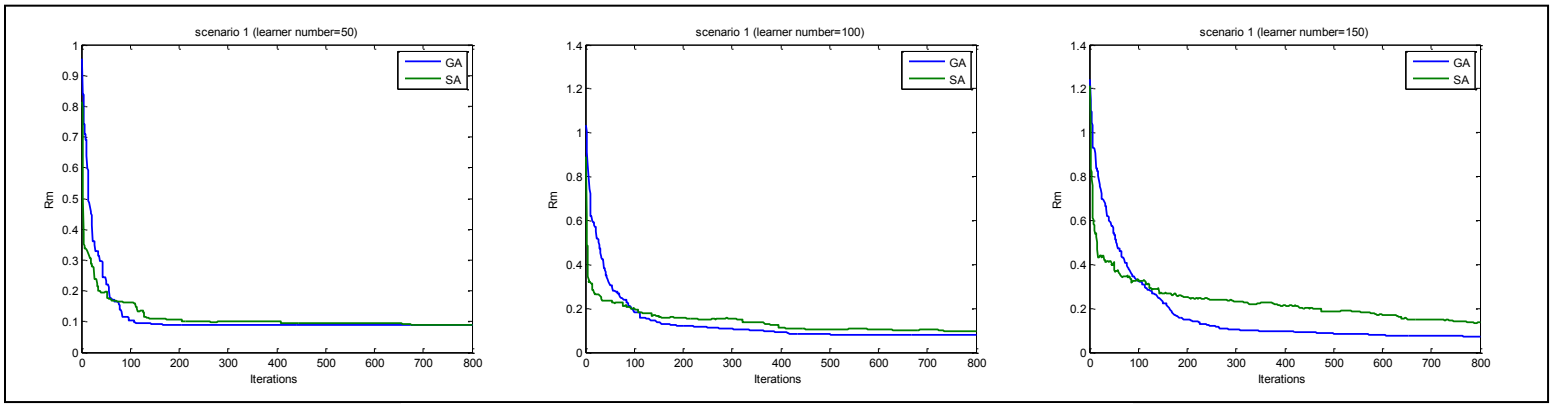

Figure.3 Comparison of the performances of GA and SA for the first scenario

Secondly, we compare the performances of GA and SA. We dismiss the restraint of max iteration times for both of them, and let them run in the condition of 50, 100 and 150 learners. The convergences are satisfactory. Take the example of algorithm running in the first scenario, as the Figure 3 has shown. GA gives the ultimate results with lower R_m value than SA, with both converging after 200 iterations. The diversities between the R_m values outputted by them are quite gradually expanding with the increase of learner numbers. So we find that the GA has better, but not distinct, efficiency for the teamwork-enhanced task allocation.

As shown in Table I, the running time of GA and SA increase in linearity, according to the number of learners. However, SA is obviously faster than GA. In addition, the running time of GA does not vary very much due to the change of crossover probability and mutation probability.

TABLE I. RUNNING TIME OF SA AND GA

Number of Learners	Running Time (in seconds)	
	SA	GA
50	6s	77s
100	13s	161s
150	19s	242s

Above all, although SA yields a little poorer solutions than GA, it is still recommended to be adopted in the Inference Service of TaaS, because it responses in a shorter time.

VI. CONCLUSION

In this paper, we introduce a new system, TaaS, for enhancing teamwork performance, which is mobile-accessible and working with current LMSs in the cloud environment. We use the KLS to accurate evaluate learners' capability. The core of the system is task allocation, which is designed for avoiding the confusion and the misunderstanding in the teamwork process, and letting the learners give full play to their talents. We describe a model of this problem, combining learners' capabilities and preferences, and tasks' difficulties. Two heuristic algorithms, GA and SA are used to solve the problem. Their algorithm details are given in Section IV. Experiments prove that both algorithms are feasible to complete the teamwork-enhanced task allocation, yielding the results satisfying our design purpose. We also compared the performances of both algorithms. Due to SA's faster running speed, we suggest that it is better to be adopted in the real implementation of TaaS.

REFERENCES

[1] T. L. Wentling, C. Waight, J. Gallaher, J. Fleur, C. Wang, C and A. Kanfer, e-learning - A Review of Literature, Knowledge and Learning Systems Group. NCSA, University of Illinois at Urbana-Champaign, Illinois, USA, 2000

[2] N. M. Rao, "Cloud Computing Through Mobile-Learning", International Journal of Advanced Computer Science and Applications, vol. 1, no. 6, pp. 42-47, 2010

[3] C. S. Saunders and M. K. Ahuja, "Are All Distributed Teams the Same? Differentiating Between Temporary and Ongoing Distributed Teams". Small Group Research. vol.37, no.6, pp.662-700, 2006.

[4] G. Sun and J. Shen, "Enhancing Teamwork Performance in Mobile Cloud-based Learning", 10th Internation Conference on Computer Support Collaborative Learning, Wisconsin, USA, June 2013.

[5] A. Y. Kolb and A. D. Kolb, "Learning styles and learning spaces: Enhancing experiential learning in higher education". Academy of Management Learning and Education. vol.4, no.2, pp193-212, 2005.

[6] D. Kolb, Experiential learning: Experience as a source of learning and development. Upper Saddle River, NJ: Prentice Hall, 1984.

[7] A. B. Kayes, D. C. Kayes and D. A. Kolb, 'Developing teams using the Kolb team learning experience'. Simulation & Gaming, vol. 36, pp. 355-363, 2005.

[8] S. A. Wheelan, Creating Effective Teams : A Guide for Members and Leaders, Sage Publications, 2005.

[9] R. Belbin, Team Roles at Work, Butterworth Heinemann, 1993.

[10] R. Lingard, "Teaching and Assessing Teamwork in Engineering and Computer Science", Proceeding of International Symposium on Engineering Education and Educational Technologies(EEET) , Orlando, USA, July 2009.

[11] E. Aronson, N. Blaney, C. Steophan, J. Sikes and M. Snapp, The jigsaw Classroom, Beverly Hills, CA, USA, 1978.

[12] J. H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, USA, 1992.

[13] J. P. Andrew, M. K. Thomas and J. N. Thomas, "Multi-heuristic Dynamic Task Allocation Using Genetic Algorithms in a Heterogeneous Distributed System", Journal of Parallel Distributed Computing, vol. 70, pp. 758-766, 2010.

[14] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing", Science, vol.220, no.4598, pp. 671-680, 1983.

[15] V. Granville, M. Krivanek and J. P. Rasson, "Simulated Annealing: a Proof of Convergence", IEEE Transaction on Pattern Analysis and Machine Intelligence, vol.16, no.6, pp. 652-656, 1994